

Tipos de Web Panels

GeneXus™

The screenshot displays a web application interface for managing attractions. It is divided into several sections:

- Attractions List:** A table listing attractions with columns for Id, Name, Country Name, Category Name, City Name, and Address. Each row includes 'UPDATE' and 'DELETE' buttons. The list includes:

Id	Name	Country Name	Category Name	City Name	Address	UPDATE	DELETE
4	Christ the Redeemer	Brazil	Monument	Rio de Janeiro			
3	Eiffel Tower	France	Monument	Paris			
7	Forbidden city	China	Tourist site	Beijing			
1	Louvre Museum	France	Museum	Paris			
6	Matisse Museum	France	Museum	Nice			
5	Smithsonian Institute	United States	Museum	Washington			
2	The Great Wall	China	Tourist site	Beijing			
- Attraction Detail View:** A form for editing an attraction. The 'Name' field is filled with 'Christ the Redeemer'. Other fields include Country Id (1), Country Name (Brazil), Category Id (2), Category Name (Monument), Photo (a small image of Christ the Redeemer), City Id (1), and City Name (Rio de Janeiro).
- Trips Table:** A table showing trips for various attractions.

Attraction Name	Country Name	Attraction Photo	Trips	New trip
Louvre Museum	France		1	New trip
The Great Wall	China		0	New trip
Eiffel Tower	France		2	New trip
Christ the Redeemer	Brazil		2	New trip
Smithsonian Institute	United States		1	New trip
Matisse Museum	France		2	New trip
Forbidden city	China		1	New trip
Trips			9	

Nos vídeos anteriores construímos do zero uma solução muito parecida com a que é construída automaticamente, pelo padrão Work With.

Assim, se compararmos em execução ambas as soluções, vemos primeiramente que diferem quanto ao desenho. É que até agora nos concentramos na lógica e deixamos de lado a User Interface, tema no qual entraremos mais tarde. Mas deixando de lado essas diferenças visuais, em ambos web panels podemos ver um grid que mostra informações das atrações turísticas, com filtros, e a possibilidade, por exemplo, de atualizar a informação.

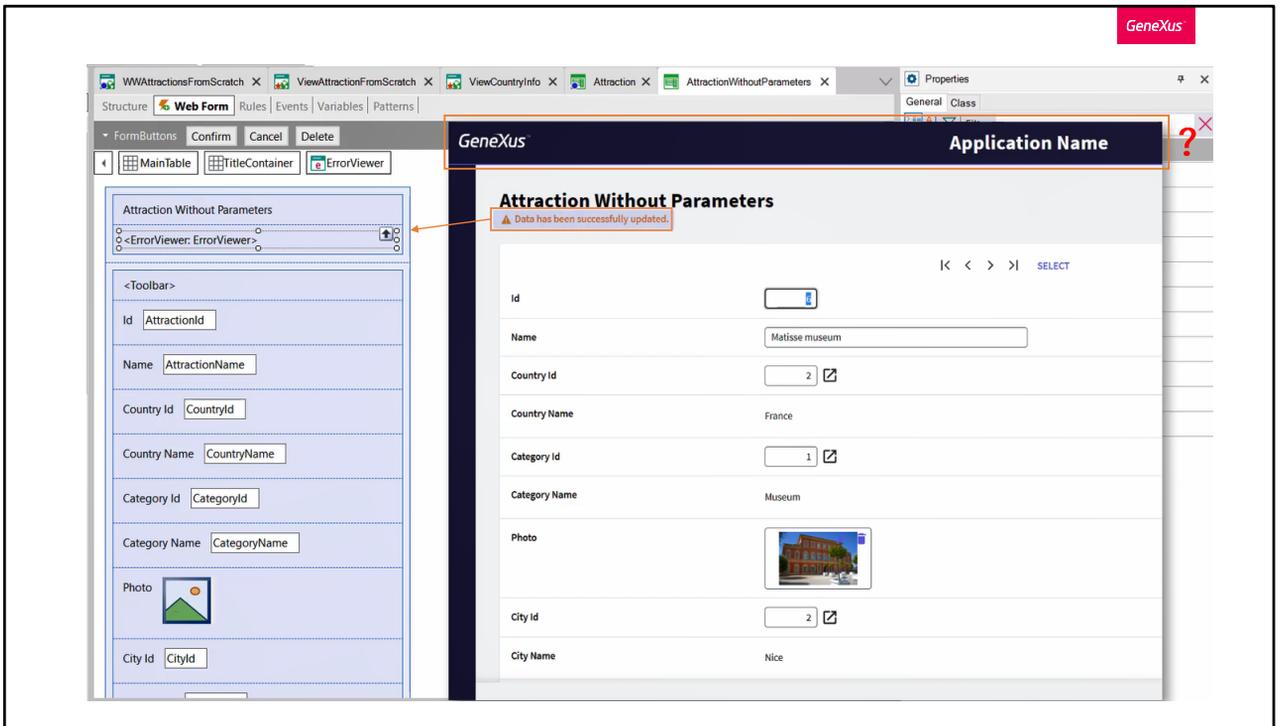
Nas duas soluções é invocada a transação em modo Update.

Algo que já pode começar a nos chamar a atenção é uma constante de todas as telas que temos navegado. Vemos que página que abrimos, página que tem esta parte.

Por exemplo, se invocarmos diretamente a transação paralela `AttractionWithoutParameters`, aqui está.

Ou, se por exemplo, agora vamos ao View de uma atração, o do pattern

primeiro ou o implementado do zero por nós depois, vemos que apesar das diferenças, seguem mantendo isto em comum.



Se agora vamos ao GeneXus ver o form deste View que nós implementamos, onde está essa parte comum?

Ou, se abrirmos a transação Attraction, ou a transação paralela que não recebe parâmetros, e vamos para seu Form, também não a vemos.

Aqui o que estamos vendo é um controle textblock com o nome da transação. Abaixo estamos vendo um controle ErrorViewer que é onde as mensagens aparecem, por exemplo, de sucesso ou fracasso. Então, temos este controle "action group" que é o que apresenta os botões de navegação e o Select. Depois, vêm os atributos e, por último, temos outro "action group" com os botões.

Onde está a área de cima?

Master Page

Web Transaction	
Style	TravelAgency
Form Layout	UnanimoTemplate
Type	Web Page
Master Page	MasterUnanimoSidebar

Web Panel: WWAttractionsFromScratch	
Name	WWAttractionsFromScratch
Description	WWAttractions From Scratch
Module/Folder	Root Module
Style	TravelAgency
Type	Web Page
Master Page	MasterUnanimoSidebar

Web Master Panel: MasterUnanimoSidebar	
Name	MasterUnanimoSidebar
Description	Master Unanimo Sidebar
Module/Folder	UI
Style	TravelAgency
Type	Master Page
Show Master Page when P	False
On session timeout	Ignore
Focus control	Use Environment property value
Cache expiration lapse	
Automatic refresh	Yes
Auto compress http traffic	Use Environment property value

Se observamos as propriedades da transação, vemos um grupo Web Transaction com quatro propriedades importantes: na primeira é atribuído o objeto com o qual será aplicado o estilo, na segunda é especificado o template que será utilizado para gerar o form, a terceira veremos a seguir, e a quarta é a que veremos agora, a propriedade Master page. Ali se indica qual será a página mestra, dentro da qual a página correspondente a este objeto transação será carregada. Vemos que nos oferece algumas possibilidades, que correspondem a todas as páginas mestras definidas, no momento, em nossa KB. Quando se cria uma KB, estas páginas são criadas para já ter algumas por padrão, mas podemos criar outras. Vemos que, por padrão, a transação foi criada associada a esta página mestra.

Se agora abrimos os outros objetos que vimos em execução, não ficaremos surpresos em descobrir que eles também têm uma página mestra, e que é esta mesma.

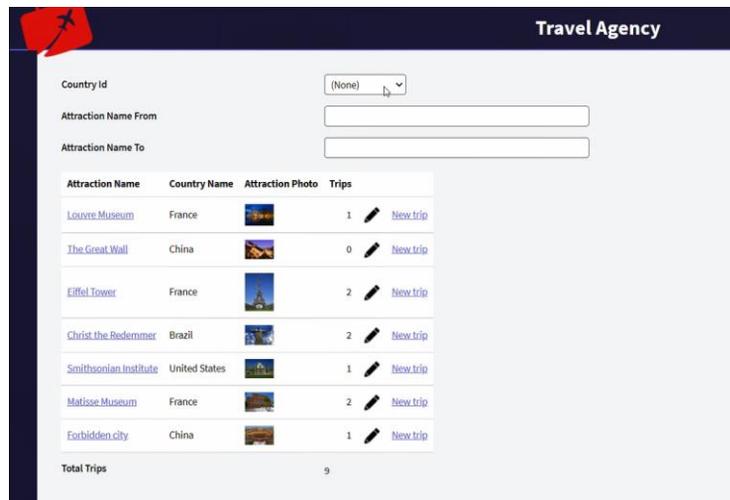
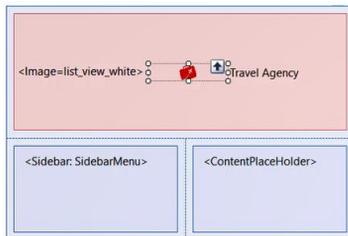
Já de início, vemos que os Web panels também contam com as mesmas quatro propriedades.

Vamos procurar esta página mestra para ver do que se trata? Se não sabemos onde se encontra, podemos procurá-la por aqui e abri-la.

Se observamos suas propriedades, vemos a de nome Style e a de nome Type, e não está mais lá a Master Page. Por quê? Porque trata-se nem mais nem menos de um web panel de um tipo especial, o tipo Master Page. Um web panel deste tipo terá um controle especial, o controle ContentPlaceHolder. Aqui é onde toda página que tenha a esta como sua página mestra, será carregada.

Então a transação será carregada neste espaço, o view, tudo o que vimos.

Master Page



Aqui vemos o controle text block que vemos em execução com o nome Application Name. Por exemplo, vamos tentar mudá-lo para Travel Agency. Aqui temos uma imagem que também podemos mudar para esta outra que previamente inserimos na KB.

Vamos testar o efeito destas mudanças em nossa aplicação. Aqui as vemos.

Outro controle que vemos neste Master Panel é um User Control, próprio do design system Unanimo, chamado Sidebar. Este controle vem por padrão e mostrará uma barra lateral suspensa na qual poderemos ter acesso a diferentes objetos.

Web Component

The image shows two screenshots from the GeneXus IDE. The left screenshot displays a 'Web Form' for 'WWAttractionsFromScratch'. It features a 'Country Id' dropdown menu, 'Attraction Name From' and 'Attraction Name To' text boxes, a grid with columns for 'Attraction Id', 'Attraction Name', 'Country Name', 'Attraction Photo', 'Trips', and a '+newTrip' button, and a 'Total Trips' label. The right screenshot shows the 'Grid's Conditions' dialog box with the following code:

```
CountryId = &CountryId
when not &CountryId.IsEmpty();

AttractionName >= &AttractionNameFrom
when not &AttractionNameFrom.IsEmpty();

AttractionName <= &AttractionNameTo
when not &AttractionNameTo.IsEmpty();
```

Below the dialog, another screenshot shows a 'ViewAttractionFromScratch' form with fields for 'Name', 'Country Id', 'Country Name', 'Category Name', 'City Name', and 'Trips'. An event handler is defined for the 'CountryName' field:

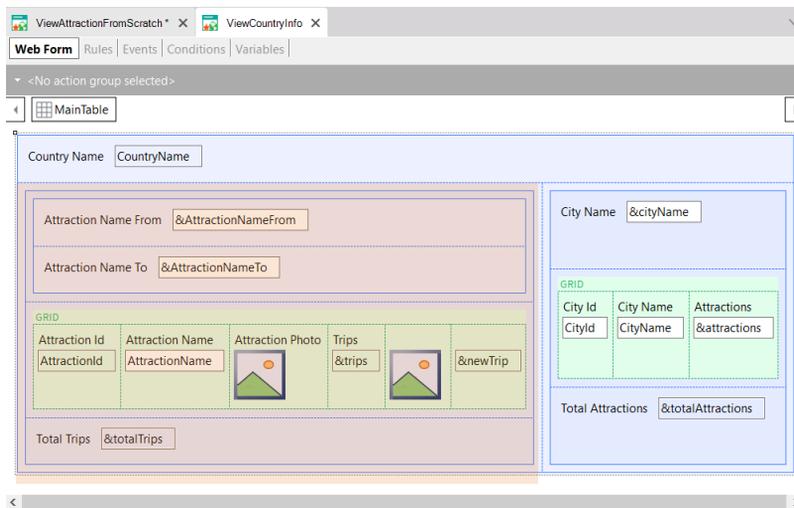
```
Event CountryName.Click
ViewCountryInfo(CountryId)
Endevent
```

Bem, com isso já vimos dois dos três tipos de telas web: a página mestra e a página comum, que é com a qual trabalhamos até agora, cada vez que criamos um web panel ou uma transação. Nos resta ver somente a página web de tipo componente.

Para isso, voltemos um pouco sobre nossos passos e recordemos o que tínhamos implementado nos vídeos anteriores. Tínhamos a imitação do Work With de atrações, em que o usuário podia filtrar por país escolhendo um valor nesta variável, que estava sendo utilizada nas conditions do grid para filtrar por esse país, se a variável não estava vazia.

Mas além disso, a partir deste painel se permitia ao usuário ver as informações de uma atração, clicando em seu nome e, por sua vez, a partir dali apresentava-se um link sobre o nome de país, para mostrar toda a informação relevante do país.

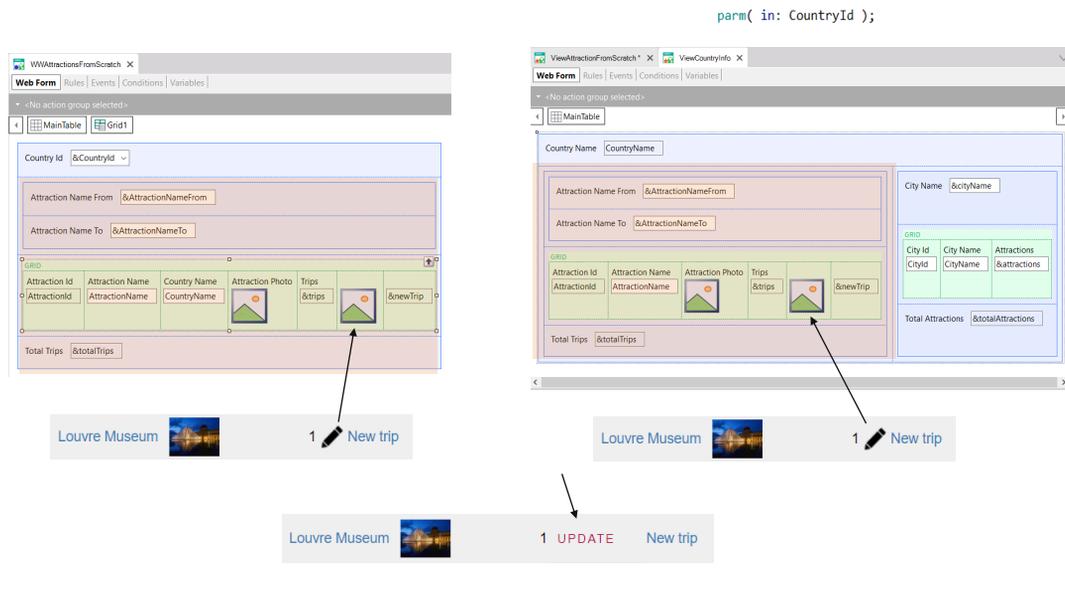
Web Component



```
param( in: CountryId );
```

Para isso chamávamos este web panel que recebia no atributo o identificador de país e mostrava seu nome, e depois, um grid idêntico ao primeiro, com os mesmos filtros por nome de atração, as mesmas colunas e ações e informações em geral e, além disso, informação das cidades.

Web Component



Vemos claramente que há uma parte deste painel que é praticamente idêntica à do outro. Suponhamos que já não queremos que a opção de Update seja oferecida com esta imagem, mas que queremos que seja como é no pattern, com a palavra UPDATE.

Teremos que substituir esta imagem por um text block em ambos os painéis. Para evitar estas duplicações e programar somente uma vez o comportamento e desenho de uma parte do panel, é que temos web panels de tipo componente.

Web Component

```
parm( in: &CountryId );
```

The screenshot displays the GeneXus IDE interface for a web form named 'CountryAttractionsInfo'. The form contains several elements: two text input fields for 'Attraction Name From' and 'Attraction Name To', a grid with columns for 'Attraction Id', 'Attraction Name', 'Attraction Photo', and 'Trips', and a 'Total Trips' field. The 'Grid1's Conditions' dialog is open, showing the following conditions:

```
CountryId = &CountryId
when not &CountryId.IsEmpty();

AttractionName >= &AttractionNameFrom
when not &AttractionNameFrom.IsEmpty();

AttractionName <= &AttractionNameTo
when not &AttractionNameTo.IsEmpty();
```

The 'Properties' window on the right shows the component's configuration:

Name	CountryAttractionsInfo
Description	Country Attractions Info
Module/Folder	Root Module
Theme	Carmine
Type	Component
URL access	No

O que no nosso caso se repete? Toda esta seção da tela e seu comportamento.

Então o que fizemos foi um Save as deste web panel, no qual unicamente retiramos a variável CountryId do form e a colocamos, em troca, como parâmetro.

O usuário já não a adicionará diretamente no form desta tela, mas será recebida de quem o chame.

Vemos que as conditions se mantêm idênticas, assim como os eventos e todo o resto. A outra mudança é que modificamos a propriedade Type, passando-a para Component. A partir de agora, este web panel poderá ser um componente de outro.

Component

The screenshot displays the GeneXus IDE interface. On the left, a 'Web Form' is shown with a 'Country Id' dropdown menu and a component placeholder labeled '<Component: CountryAttractionsInfo>'. The 'Properties' window on the right shows the configuration for 'Web Component: Component1':

Property	Value
Control Name	Component1
Object	CountryAttractionsInfo
Parameters	&CountryId

Below the properties, there are sections for 'Cell information' and 'Row information'. The 'Cell information' section includes: Cell Control Name, Cell Class, Horizontal Alignment (Default), and Vertical Alignment (Default). The 'Row information' section includes: Row Height and Row Class.

On the right, a preview window titled 'Travel Agency' shows the rendered form. The 'Country Id' dropdown menu is highlighted with a red box and contains the value 'France'. Below it are input fields for 'Attraction Name From' and 'Attraction Name To'. A table lists attractions with columns for 'Attraction Name', 'Country Name', 'Attraction Photo', and 'Trips'. The table data is as follows:

Attraction Name	Country Name	Attraction Photo	Trips
Louvre Museum	France		1 New trip
The Great Wall	China		0 New trip
Eiffel Tower	France		2 New trip
Christ the Redeemer	Brazil		2 New trip
Smithsonian Institute	United States		1 New trip
Museum of Modern Art	France		2 New trip
Forbidden city	China		1 New trip

The total number of trips is 9.

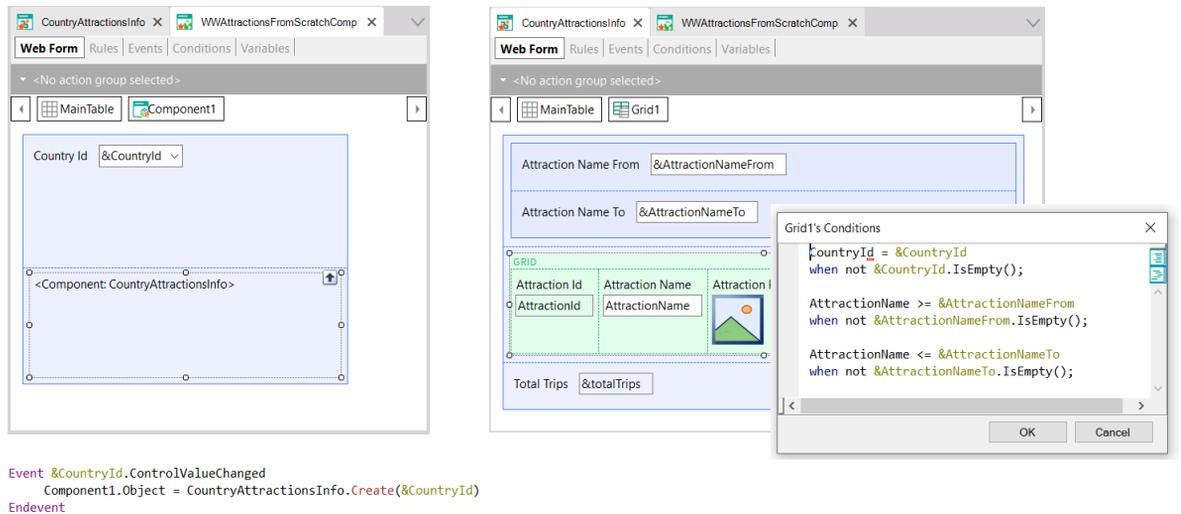
Então, a próxima coisa que fizemos foi um save as do nosso painel original, e substituímos todos esses controles que agora colocamos no web panel componente, por um controle de tipo componente. Evidentemente, eliminamos (aqui deixamos comentados) todos os eventos que agora estarão no componente.

Ao colocar um controle deste tipo, estamos dizendo que ali dentro, deverá ser carregado e executado o que especificarmos. Temos que dizer-lhe que nesse componente seja criada uma instância do web component que acabamos de mostrar, CountryAttractionsInfo. Podemos fazer de forma estática, indicando-o assim nas propriedades, onde aqui indicamos qual será o objeto de tipo componente, e aqui o parâmetro enviado. Vamos testar.

Vemos exatamente o mesmo. Se filtrarmos por nome de atração... está funcionando, perfeito.

Agora, o que acontece se queremos filtrar por país? Nada acontece. Por quê?

Component



The screenshot shows two views of a web form in GeneXus. The left view shows a dropdown menu for 'Country Id' with the value '&CountryId'. The right view shows a grid with columns for 'Attraction Id', 'Attraction Name', and 'Attraction Image', and a 'Total Trips' column. The grid is filtered based on the selected country. A dialog box titled 'Grid1's Conditions' is open, showing the following code:

```
CountryId = &CountryId
when not &CountryId.IsEmpty();

AttractionName >= &AttractionNameFrom
when not &AttractionNameFrom.IsEmpty();

AttractionName <= &AttractionNameTo
when not &AttractionNameTo.IsEmpty();
```

Below the grid, the event handler for the dropdown menu is defined:

```
Event &CountryId.ControlValueChanged
Component1.Object = CountryAttractionsInfo.Create(&CountryId)
Endevent
```

A variável CountryId está em um painel diferente do que o grid de atrações pelo qual se filtra. Aqui o que temos que fazer é utilizar o evento ControlValueChanged, que captura o momento em que se modifica o valor do controle variável &CountryId, e pedir que se crie uma nova instância do componente, passando-lhe agora o novo valor da variável.

Vamos testar

Atualizamos. Tentamos filtrar por França. Perfeito. E, lá dentro, por nome de atração. Perfeito também.

Component

```
parm( in: CountryId );
```

The screenshot displays the GeneXus IDE interface. On the left, the design view shows a web form with a 'CountryName' input field, a 'Component2' placeholder, and a 'CountryAttractionsInfo' component. This component contains a grid with columns 'City Id', 'City Name', and 'Attractions', and a 'Total Attractions' field. On the right, the Properties window for 'Web Component: Component2' is open, showing the following details:

Control Name	Component2
Object	CountryAttractionsInfo
Parameters	CountryId
Cell information	
Cell Control Name	

Event Grid2.Load

```
&attractions = Count(AttractionName)
&totalAttractions = &totalAttractions + &attractions
endevent
```

Event Grid2.Refresh

```
&totalAttractions = 0
Endevent
```

E, claro, fizemos algo similar com o painel que mostrava as informações do país.

Fizemos um Save as deste painel, no qual substituímos toda esta seção por um componente que será carregado com este painel.

Assim, no novo painel o CountryId não é variável, mas é recebido por parâmetro, e por isso podemos criar a instância do componente de forma estática, uma única vez dentro da execução deste web panel, passando-lhe aqui o parâmetro que neste caso vem no atributo recebido na regra parm.

Observemos que do painel original, removemos os controles e a programação de eventos associados às atrações, e só deixamos as das cidades. Testemos em execução.

Para isso, invoquemos este painel e não o anterior.

Travel Agency

Country Name

Attraction Name From

Attraction Name To

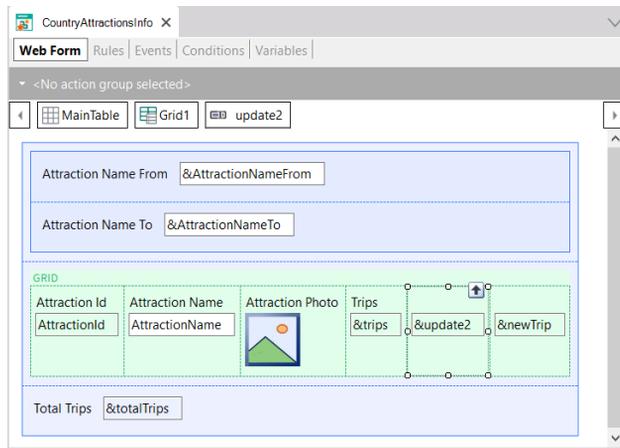
City Name

Attraction Name	Attraction Photo	Trips	update
Eiffel Tower		2	 New trip
Louvre Museum		1	 New trip
Matisse Museum		2	 New trip
Total Trips		5	

City Id	City Name	Attractions
1	Paris	2
2	Nice	1
Total Attractions		3

Agora, modifiquemos a imagem para realizar o UPDATE, e coloquemos em seu lugar o texto UPDATE.

Web Component



```

Event Grid1.Load
    &trips = Count(TripDate)
    &totalTrips = &totalTrips + &trips
Endevent

Event Grid1.Refresh
    &totalTrips = 0
Endevent

Event Start
    &update2 = "UPDATE"
    &newTrip = "New trip"
Endevent

Event &update2.Click
    Attraction(trnMode.Update, AttractionId)
Endevent

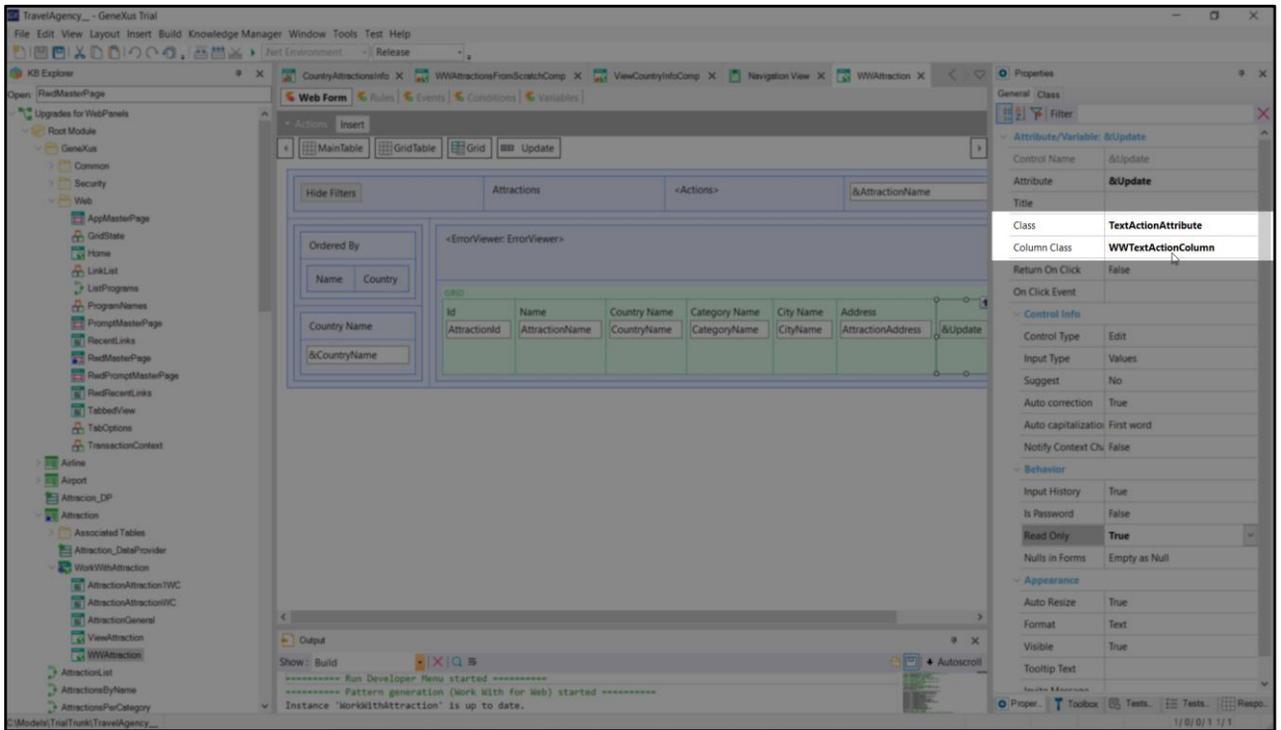
```

Para isso, no web panel componente criamos uma variável de tipo Character de 20. A inserimos no grid. No evento Start atribuímos-lhe o valor UPDATE, que é o que queremos que o usuário veja. Eliminamos a atribuição de uma imagem à variável que tínhamos, porque vamos eliminá-la do grid.

E, agora, vamos tirar o título da nova variável e torná-la Read only.

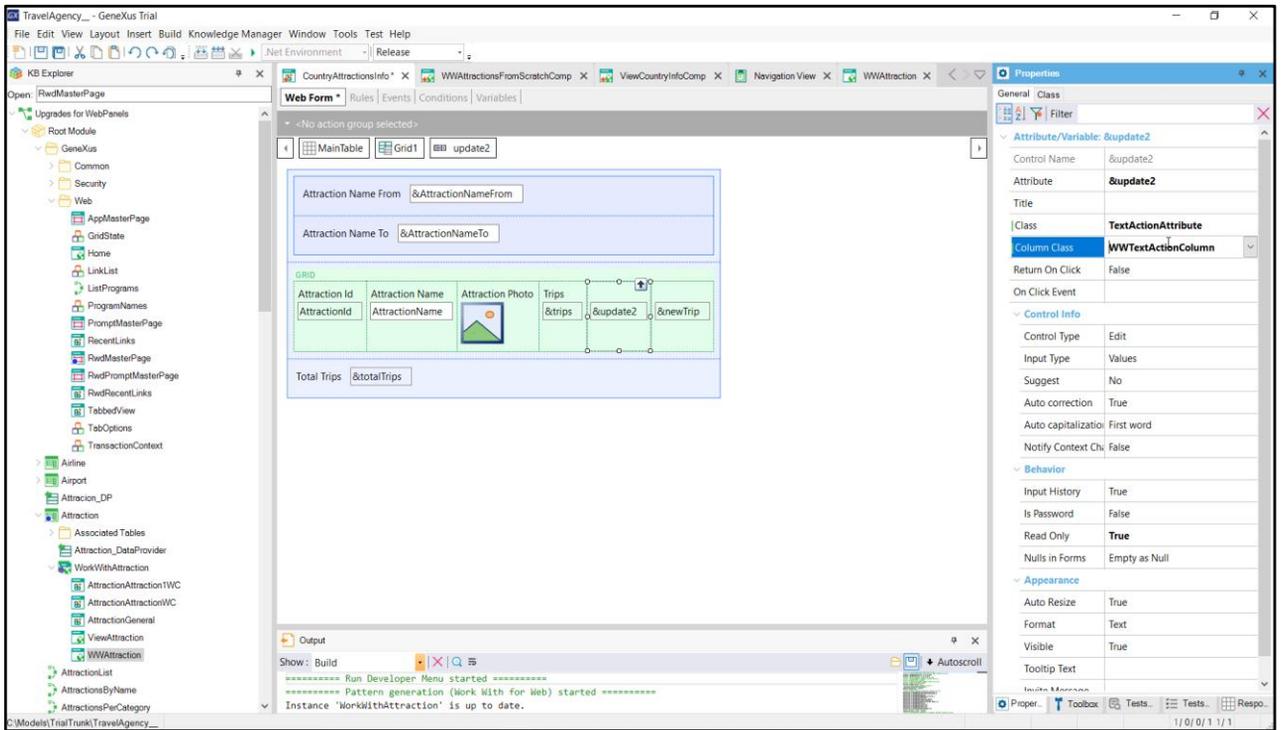
Vamos tentar executar. Nos indica que o click não é um evento válido. É que esquecemos de modificá-lo para que seja o click da nova variável e não da velha.

Agora sim... executemos.



Se vamos observar como ficou no pattern, por que aqui tem este desenho tão mais bonito?

Localizemos as propriedades do controle no web panel gerado pelo pattern. Vejamos estas duas: Class e Column Class.



E vemos os valores que essas propriedades têm para o nosso controle, o que nós inserimos manualmente em nosso web panel. São diferentes. Vamos atribuir-lhes as mesmas que as do pattern. E testemos.

The screenshot shows a web application titled "Travel Agency" with a dark blue header. On the left, there is a sidebar with a red location pin icon. The main content area is divided into several sections:

- Country Name:** A dropdown menu currently showing "France".
- Attraction Name From:** An empty text input field.
- Attraction Name To:** An empty text input field.
- City Name:** An empty text input field.
- Attraction List Table:**

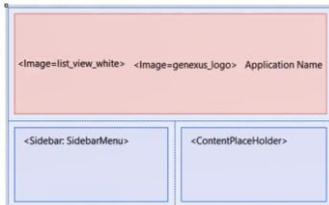
Attraction Name	Attraction Photo	Trips		
Eiffel Tower		2	UPDATE	New trip
Louvre Museum		1	UPDATE	New trip
Matisse Museum		2	UPDATE	New trip
Total Trips		5		
- City Attractions Table:**

City Id	City Name	Attractions
1	Paris	2
2	Nice	1
Total Attractions		3

Aqui o vemos. Com isto já começamos a entender como se manipula o desenho de nossas telas.

Types of Web Panels

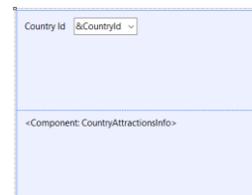
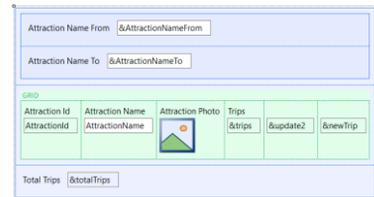
Master Page



Web Page



Component



Resumindo: Vimos três tipos de Web panels que estão relacionados entre si.

O do tipo Master Page, que oferece uma estrutura comum a todas as páginas da aplicação ou de uma parte dela, para não ter que repetir o mesmo cada vez, para cada página. Por exemplo, os menus costumam ir aqui. Este objeto tem a particularidade de conter em seu form um controle especial, o ContentPlaceHolder, onde as páginas web serão carregadas.

O do tipo Web Page, que é o que vínhamos estudando, que poderá ter associada uma Master Page determinada e só uma, visto que será carregado no ContentPlaceHolder desta.

E o do tipo Component, que serve justamente para reutilizar desenho e programação em diferentes objetos. Para poder fazer com que um objeto definido como Web panel de tipo componente se carregue dentro de outro objeto web, utiliza-se o controle de tipo component, que poderá ser carregado estática ou dinamicamente.

Quanto mais componentes identificarmos e utilizarmos, melhor qualidade terá a aplicação resultante.

Claro, há muito mais para estudar sobre este tema (por exemplo, o que acontece com a execução dos eventos em um panel com componentes, como se atualiza um componente, etc.). Por agora, com isto é mais do que suficiente.

GeneXus[™]

training.genexus.com
wiki.genexus.com